Documentação Técnica do Projeto NutriX Pro AI

1. Arquitetura Geral e Filosofia de Design

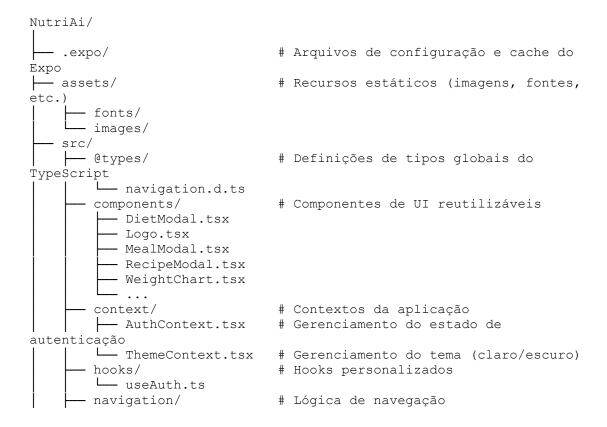
O NutriX pro AI é um aplicativo móvel multiplataforma construído com base em uma arquitetura modular e reativa, centrada no usuário. A filosofia de design visa a simplicidade e a personalização, oferecendo uma experiência intuitiva para o rastreamento nutricional e o acesso a informações de saúde.

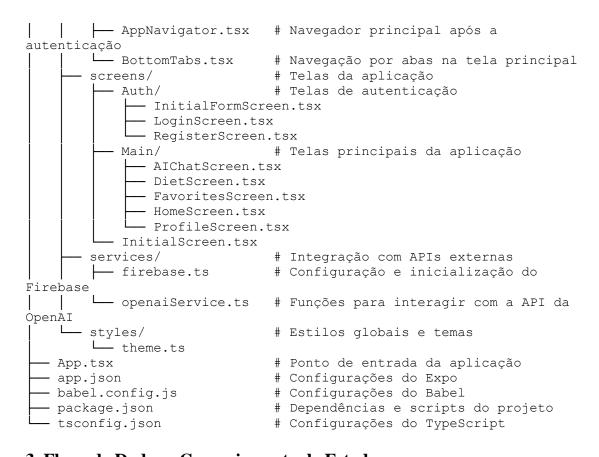
A arquitetura do software segue os seguintes princípios:

- **Componentização:** A interface do usuário é construída a partir de componentes reutilizáveis, garantindo consistência visual e facilitando a manutenção.
- **Separação de Responsabilidades (SoC):** A lógica de negócios, a interface do usuário e os serviços de backend são separados em módulos distintos, promovendo um código mais limpo e organizado.
- **Estado Centralizado:** O estado global da aplicação, como informações de autenticação e tema, é gerenciado através da Context API do React, simplificando o compartilhamento de dados entre os componentes.
- Tipagem Estática: O uso do TypeScript em todo o projeto garante a segurança dos tipos, reduzindo a ocorrência de erros em tempo de execução e melhorando a legibilidade do código.

2. Estrutura Detalhada do Projeto

A estrutura de diretórios do projeto foi projetada para promover a escalabilidade e a organização do código-fonte:





3. Fluxo de Dados e Gerenciamento de Estado

O gerenciamento de estado no NutriX AI é realizado principalmente através da **Context API do React**. Dois contextos principais governam o estado global da aplicação:

- AuthContext: Responsável por gerenciar o estado de autenticação do usuário. Ele armazena as informações do usuário logado e fornece funções para login, logout e registro. O hook useAuth simplifica o acesso a este contexto a partir de qualquer componente.
- ThemeContext: Gerencia o tema da aplicação (claro ou escuro), permitindo a alteração dinâmica da aparência da interface do usuário.

Para o estado local dos componentes, o hook usestate do React é amplamente utilizado, mantendo a simplicidade e a reatividade da interface.

4. Integração com APIs

4.1. Firebase

O Firebase é utilizado como o principal serviço de backend, fornecendo:

- **Autenticação:** O Firebase Authentication é usado para o gerenciamento de usuários, incluindo cadastro, login e logout com e-mail e senha.
- **Banco de Dados:** O Firestore é o banco de dados NoSQL utilizado para armazenar os dados do usuário, como:

- Perfis de Usuário: Informações pessoais, objetivos, restrições alimentares e estilo de vida.
- o **Refeições:** Registros diários de refeições e estimativas de calorias.
- o Receitas Favoritas: Receitas salvas pelo usuário, organizadas em pastas.
- Histórico de Peso: Registros periódicos do peso do usuário para o acompanhamento da evolução.

4.2. OpenAI (GPT API)

A integração com a API da OpenAI é realizada através do openaiService.ts e é fundamental para as funcionalidades de inteligência artificial do aplicativo:

- Estimativa de Calorias: Quando um usuário registra uma refeição, uma requisição é enviada à API do GPT com a descrição dos alimentos. A IA analisa o texto e retorna uma estimativa das calorias, que é então armazenada no Firestore.
- **Geração de Receitas:** Os usuários podem solicitar receitas com base em ingredientes específicos ou preferências alimentares. A API do GPT gera uma receita completa, incluindo ingredientes e instruções.
- Chat Inteligente: O AIChatScreen utiliza a API do GPT para fornecer um assistente virtual que responde a perguntas sobre nutrição e oferece sugestões personalizadas.

5. Navegação

A navegação no NutriX AI é gerenciada pela biblioteca React Navigation. A estrutura de navegação é dividida em duas partes principais:

- 1. Navegador de Autenticação: Uma pilha de navegação (StackNavigator) que controla o fluxo de autenticação, incluindo as telas de login, registro e o formulário inicial de coleta de dados.
- 2. **Navegador Principal:** Após a autenticação, o usuário é direcionado para o AppNavigator, que consiste em uma navegação por abas (BottomTabNavigator). As abas principais são:
 - o **Home:** Dashboard com o resumo diário.
 - o **Dieta:** Tela para registro e visualização de refeições.
 - o **Chat AI:** Acesso ao assistente virtual.
 - o **Favoritos:** Gerenciador de receitas salvas.
 - o **Perfil:** Configurações do usuário e informações pessoais.

6. Estilização e Temas

A aplicação suporta temas claro e escuro, implementados através do ThemeContext e de folhas de estilo dinâmicas. O arquivo src/styles/theme.ts define as cores e fontes para cada tema. Os componentes utilizam o hook useContext para acessar o tema atual e aplicar os estilos correspondentes, garantindo uma experiência visual consistente e personalizável.

7. Scripts e Execução

O arquivo package.json define os seguintes scripts para o desenvolvimento e execução do projeto:

- "start": Inicia o servidor de desenvolvimento do Expo.
- "android": Inicia o aplicativo em um emulador ou dispositivo Android.
- "ios": Inicia o aplicativo em um emulador ou dispositivo iOS.
- "web": Inicia o aplicativo em um navegador web.

Para executar o projeto, é necessário ter o Node.js e o Expo CLI instalados. Após a clonagem do repositório e a instalação das dependências com npm install, o projeto pode ser iniciado com npx expo start.

8. Funcionalidades Principais

O aplicativo oferece um conjunto de funcionalidades para auxiliar o usuário em sua jornada de alimentação saudável:

- **Autenticação de Usuário:** Sistema de cadastro e login com e-mail e senha, utilizando o Firebase Authentication.
- **Formulário Inicial:** Coleta de informações do usuário para personalizar a experiência, incluindo dados pessoais, objetivos, estilo de vida e restrições alimentares
- **Dashboard** (**Tela Inicial**): Apresenta um resumo do consumo diário de calorias e a evolução do peso do usuário em gráficos.
- **Registro de Refeições:** Permite ao usuário registrar suas refeições, com a opção de estimar as calorias com o auxílio da IA.
- Chat com IA: Um chat inteligente que utiliza a API do GPT para responder a dúvidas sobre nutrição, sugerir receitas e adaptações na dieta.
- **Gerenciador de Receitas Favoritas:** Os usuários podem salvar e organizar suas receitas favoritas em pastas personalizadas.
- **Dicas Personalizadas:** Geração de dicas de nutrição e bem-estar com base no perfil e objetivos do usuário.

9. Configuração do Ambiente e Execução

Para executar o projeto, siga os passos abaixo:

1. **Instalação de Dependências:** Certifique-se de ter o Node.js instalado. Em seguida, clone o repositório e instale as dependências necessárias executando o seguinte comando no terminal:

Bash

npm install

2. **Configuração de Variáveis de Ambiente:** O projeto utiliza um arquivo .env para armazenar as chaves de API do Firebase e da OpenAI. Certifique-se de criar este arquivo na raiz do projeto e adicionar as seguintes variáveis:

- o OPENAI API KEY
- o FIREBASE API KEY
- o FIREBASE AUTH DOMAIN
- o FIREBASE PROJECT ID
- o FIREBASE STORAGE_BUCKET
- o FIREBASE MESSAGING_SENDER_ID
- o FIREBASE APP ID
- 3. **Execução do Projeto:** Para iniciar o aplicativo, utilize o Expo. Execute o comando abaixo no terminal e, em seguida, escaneie o QR code com o aplicativo Expo Go no seu celular ou utilize um emulador:

Bash

npx expo start

10. Observações Adicionais

- O projeto utiliza a biblioteca react-native-progress-steps para criar o formulário inicial de forma progressiva.
- A navegação do aplicativo é gerenciada pelo React Navigation, com uma estrutura de navegação em abas (BottomTabs) para as telas principais e uma navegação em pilha (Stack.Navigator) para as demais telas, como login, cadastro e o formulário.
- Os estilos são criados de forma dinâmica com base no tema (claro ou escuro) selecionado pelo usuário, utilizando a Context API do React.
- A integração com a API da OpenAI é realizada através do serviço openaiService.ts, que contém funções para estimar calorias, gerar receitas e interagir com o chat.